Comparative Evaluation of Anomaly Detection Techniques for Sequence Data

Varun Chandola, Varun Mithal, and Vipin Kumar

Abstract

We present a comparative evaluation of a large number of anomaly detection techniques on a variety of publicly available as well as artificially generated data sets. Many of these are existing techniques while some are slight variants and/or adaptations of traditional anomaly detection techniques to sequence data. The specific contributions of this paper are as follows: (i). This evaluation facilitates understanding of the relative strengths and weaknesses of different techniques. Through careful experimentation, we illustrate that the performance of different techniques is dependent on the nature of sequences, and the nature of anomalies in the sequences. No one technique outperforms all others. For most techniques we also identify some data sets on which they perform very well, and some on which they perform poorly. (ii). We investigate variants that have not been tried before. For example, we evaluate a k nearest neighbor based technique that performs better than a clustering based technique that was proposed for sequences. Also, we propose FSA-z, a variant of an existing Finite State Automaton (FSA) based technique, which performs consistently superior to the original FSA based technique. (iii). We propose a novel way of generating artificial sequence data sets to evaluate anomaly detection techniques. (iv). We characterize the nature of normal and anomalous test sequences, and associate the performance of each technique to one or more of such characteristics.

1 Introduction

Sequence data is found in a wide variety of application domains such as intrusion detection, bio-informatics, etc. Hence anomaly detection for sequence data is an important topic of research. There is extensive work on anomaly detection techniques [1, 11, 14], but most of these techniques look for individual objects that are different from normal objects. These techniques do not take the sequence aspect of the data into consideration. For example, consider the set of user command sequences shown in Table 1. Clearly the sequence S_5 is anomalous, even though each command in the sequence by itself is normal.

Several anomaly detection techniques for symbolic sequences have been proposed in diverse application domains such as intrusion detection, proteomics, and aircraft safety

S_1	$login, pwd, mail, ssh, \ldots, mail, web, logout$
S_2	$login, pwd, mail, web, \ldots, web, web, web, logout$
S_3	$login, pwd, mail, ssh, \ldots, mail, web, web, logout$
S_4	$login, pwd, web, mail, ssh, \ldots, web, mail, logout$
S_5	$\mathbf{login}, \mathbf{pwd}, \mathbf{login}, \mathbf{pwd}, \mathbf{login}, \mathbf{pwd}, \ldots, \mathbf{logout}$

Table 1: Sequences of User Commands

Application Domains	Kernel Based Techniques	Window Based Techniques	Markovian Techniques		
			Fixed	Variable	Sparse
Intrusion Detection		[7],[12], [8],[10]	[8],[9], [20],[16], [15],[19]		[8], [6]
Proteomics				[24]	
Flight Safety	[4]		[23]		

Table 2: Anomaly Detection Techniques for Symbolic Sequences.

as shown in Table 2. We group such techniques into following three categories: **kernel based**, **window based**, and **Markovian** techniques. Kernel based techniques use a similarity measure to compute similarity between sequences. For example, a clustering based technique that uses *normalized longest common subsequence* as the similarity measure [4], has been proposed in aircraft safety domain. Window based techniques, e.g., STIDE [12], extract fixed length windows from a sequence and assign an anomaly score to each window. Markovian techniques assign a probabilistic anomaly score to each event conditioned on its history, using sequence modeling techniques. Examples of such techniques are *Finite State Automata* (FSA) [19], *Hidden Markov Models* (HMM) [20], and *Probabilistic Suffix Trees* (PST) [24].

As is evident from Table 2, most of the existing techniques for sequence anomaly detection have been tried in only one domain with no comparative evaluation. We are aware of only one work [8] that compared four techniques (namely, STIDE, t-STIDE, which is a variant of STIDE, HMM based, and RIPPER based) on 6 different data sets, all of which were from system call intrusion detection domain.

1.1 Our Contributions

We present a comparative evaluation of a large number of anomaly detection techniques on a variety of publicly available as well as artificially generated data sets. Many of these are existing techniques while some are slight variants and/or adaptations of traditional anomaly detection techniques to sequence data. The specific contributions of this paper are as follows:

- This evaluation facilitates understanding of the relative strengths and weaknesses of different techniques. Through careful experimentation, we illustrate that the performance of different techniques is dependent on the nature of sequences, and the nature of anomalies in the sequences. No one technique outperforms all others. For most techniques we also identify some data sets on which they perform very well, and some on which they perform poorly.
- We investigate variants that have not been tried before. Under kernel based techniques, we evaluate a k nearest neighbor based technique that performs better than a clustering based technique that was proposed for sequences [4]. Under Markovian techniques, we propose FSA-z, a variant of an existing *Finite State Automaton* (FSA) based technique, which performs consistently superior to the original FSA based technique [19].
- We propose a novel way of generating artificial sequence data sets to evaluate anomaly detection techniques.
- We characterize the nature of normal and anomalous test sequences, and associate the performance of each technique to one or more of such characteristics.

2 Problem Statement

The objective of the techniques evaluated in this paper can be stated as follows:

Definition 1 *Given a set of n training sequences, $\mathbf{S}$, and a set of m test sequences $\mathbf{S^T}$, find the anomaly score $A(S_q)$ for each test sequence $S_q \in \mathbf{S^T}$, with respect to $\mathbf{S}$.*

All sequences consist of events that correspond to a finite alphabet, Σ. The length of sequences in $\mathbf{S}$ and sequences in $\mathbf{S^T}$ might or might not be equal in length. The training database $\mathbf{S}$ is assumed to contain only normal sequences, and hence the techniques operate in a semi-supervised setting [25]. In Section 8, we discuss how the techniques can be extended to unsupervised setting, where $\mathbf{S}$ can contain both normal and anomalous sequences.

3 Anomaly Detection Techniques for Sequences

We evaluated a variety of techniques that can be grouped into following three categories:

3.1 Kernel Based Techniques

Kernel based techniques make use of pairwise similarity between sequences. In the problem formulation stated in Definition 1 the sequences can be of different lengths, hence simple measures such as *Hamming Distance* cannot be used. One possible measure is the normalized length of *longest common subsequence* between a pair of sequences. This similarity between two sequences S_i and S_j, is computed as:

$$nLCS(S_i, S_j) = \frac{|LCS(S_i, S_j)|}{\sqrt{|S_i||S_j|}} \quad (1)$$

Since the value computed above is between 0 and 1, $nLCS(S_i, S_j)$ can be used to represent distance between S_i and S_j [25]. Other similarity measures can be used as well, for e.g., the *spectrum kernel* [17]. We use $nLCS$ in our experimental study, since it was used in [4] in detecting anomalies in sequences and appears promising.

3.1.1 Nearest Neighbors Based (kNN)

In the nearest neighbor scheme (kNN), for each test sequence $S_q \in \mathbf{S^T}$, the distance to its k^{th} nearest neighbor in the training set $\mathbf{S}$ is computed. This distance becomes the anomaly score $A(S_q)$ [25, 21].

A key parameter in the algorithm is k. In our experiments we observe that the performance of kNN technique does not change much for $1 \leq k \leq 8$, but the performance degrades gradually for larger values of k.

3.1.2 Clustering Based (CLUSTER)

This technique clusters the sequences in $\mathbf{S}$ into a fixed number of clusters, c, using CLARA [13] k-medoids algorithm. The test phase involves measuring the distance of every test sequence, $S_q \in \mathbf{S^T}$, with the medoid of each cluster. The distance to the medoid of the closest cluster becomes the anomaly score $A(S_q)$.

The number of clusters, c, is a key parameter for this technique. In our experiments we observed that the performance of CLUSTER improved as c was increased from 2 onwards, but stabilized for values greater than 32. Generally, if the normal data set can be well represented using c clusters, CLUSTER will perform well for that value of c.

3.2 Window Based Technique (t-STIDE)

Window based techniques try to localize the cause of anomaly in a test sequence, within one or more windows, where a window is a fixed length subsequence of the test sequence. One such technique called *Threshold Sequence Time-Delay Embedding* (t-STIDE) [8] uses a sliding window of fixed size k to extract k-length windows from the training sequences in $\mathbf{S}$. The count of each window occurring in $\mathbf{S}$ is maintained. During testing, k-length windows

are extracted from a test sequence S_q. Each such window ω_i is assigned a likelihood score $P(\omega_i) = \frac{f(\omega_i)}{f(*)}$, where $f(\omega_i)$ is the frequency of occurrence of window ω_i in $\mathbf{S}$, and $f(*)$ is the total number of k length windows extracted from $\mathbf{S}$.

For the test sequence S_q, $|S_q| - k + 1$ windows are extracted, and a likelihood score vector of length $|S_q| - k + 1$ is obtained. This score vector can be combined in multiple ways to obtain $A(S_q)$, as discussed in Section 3.4.

3.3 Markovian Techniques

Such techniques estimate the conditional probability for each symbol in a test sequence S_q conditioned on the symbols preceding it. Most of the techniques utilize the *short memory* property of sequences [22]. This property is a higher-order Markov condition which states that for a given sequence $S = \langle s_1, s_2, \ldots s_{|S|} \rangle$, the conditional probability of occurrence of a symbol s_i is given as:

$$P(s_i|s_1 s_2 \ldots s_{i-1}) \approx P(s_i|s_k s_{k+1} \ldots s_{i-1}) \quad (2)$$

for some $k \geq 1$. In the following, we investigate four Markovian techniques. Each one of them computes a vector of scores, each element of which corresponds to the conditional probability of observing a symbol, as defined in Equation 2. This score vector is then combined to obtain $A(S_q)$ using techniques discussed in Section 3.4.

3.3.1 Finite State Automata Based Techniques (FSA and FSA-z)

A fixed length Markovian technique (FSA) [19] determines the probability $P(s_{qi})$ of a symbol s_{qi}, conditioned on a fixed number of preceding symbols[1]. The approach employed by FSA uses a *Finite State Automaton* to estimate the conditional probabilities.

FSA extracts $(n + 1)$ sized subsequences from the training data $\mathbf{S}$ using a sliding window. Each node in the automaton constructed by FSA corresponds to a unique subsequence of n symbols that form the first n symbols of such $n + 1$ length subsequences. An edge exists between a pair of nodes, N_i and N_j in the FSA, if N_i corresponds to states $s_{i1} s_{i2} \ldots s_{in}$ and N_j corresponds to states $s_{i2} s_{i3} \ldots s_{in} s_{jn}$. At every state of the FSA two quantities are maintained. One is the number of times the n length subsequence corresponding to the state is observed in $\mathbf{S}$. The second quantity is a vector of frequencies corresponding to number of times different edges emanating from this state are observed. Using these two quantities, the conditional probability for a symbol, given preceding n symbols, can be determined.

During testing, the automaton is used to determine a likelihood score for every $n+1$ subsequence extracted from test sequence S_q which is equal to the conditional probability associated with the transition from the state corresponding to first n symbols to the state corresponding to the last n symbols. If there is no state in the automaton corresponding to the first n symbols, the subsequence is ignored.

FSA-z We propose a variant of FSA technique, in which if there is no state corresponding to the first n symbols of a $n + l$ subsequence, we assign a low score (e.g. 0) to that subsequence, instead of ignoring it. The intuition behind assigning a low score to non-existent states is that anomalous test sequences are more likely to contain such states, than normal test sequences. While FSA ignores this information, we utilize it in FSA-z.

For both FSA and FSA-z techniques, the value of n is a critical parameter. Setting n to be very low (≤ 3) or very high (≥ 10), results in poor performance. The best results were obtained for $n = 5$.

3.3.2 Probabilistic Suffix Trees (PST)

A PST is a tree representation of a variable-order markov chain [24]. It estimates the probability, $P(s_{qi})$, of a symbol s_{qi}, in the test sequence, S_q, conditioned on a variable number of previously observed symbols. (variable markov models). We evaluate one such technique (PST), proposed by [24] using *Probabilistic Suffix Trees* [22]. In the training phase, a PST is constructed from the sequences in $\mathbf{S}$. The depth of a fully constructed PST is equal to the length of longest sequence in $\mathbf{S}$. For anomaly detection, it has been shown that the PST can be pruned significantly without affecting their performance. The pruning can be done by limiting the maximum depth of the tree to a threshold, L, or by applying thresholds to the empirical probability of a node label, $MinCount$, or to the conditional probability of a symbol emanating from a given node, $PMin$.

It should be noted that if the thresholds $MinCount$ and $PMin$ are not applied, the PST based technique is equivalent to FSA technique with $n = L$ and $l = 1$. When the two thresholds are applied, the events are conditioned on the maximum length suffix, with maximum length L, that exists in the PST.

For testing, the PST assigns a likelihood score to each event s_{qi} of the test sequence S_q as equal to the probability of observing symbol s_{qi} after the longest suffix of $s_{q1} s_{q2} \ldots s_{qi-1}$ that occurs in the PST.

3.3.3 Sparse Markovian Technique (RIPPER)

Sparse Markovian techniques are more flexible than variable Markovian techniques, in the sense that they estimate the conditional probability of s_{qi} based on a subset of symbols within the preceding k symbols, which are not necessarily contagious or immediately preceding to s_{qi}. In other words the symbols are conditioned on a sparse history.

[1] A more general formulation that determines probability of l symbols conditioned on a fixed number of preceding n symbols is discussed in [19].

Lee et al [16] use RIPPER classifier to build such sparse models. In this approach, a sliding window is applied to the training data $\mathbf{S}$ to obtain k length windows. The first $k-1$ positions of these windows are treated as $k-1$ categorical attributes, and the k^{th} position is treated as a target class. RIPPER [5] is used to learn rules that can predict the k^{th} symbol given the first $k-1$ symbols. To ensure that there is no symbol that occurs very rarely as the target class, the training sequences are over-sampled.

For testing, k length subsequences are extracted from each test sequence S_q using a sliding window. For any subsequence, the first $k-1$ events are classified using the classifier learnt in the training phase and the prediction is compared to the k^{th} symbol. RIPPER also assigns a confidence score associated with the classification, denoted as $conf(s_{qi})$. Lee et al assign the likelihood score of symbol s_{qi} is assigned as follows:

- For a correct classification, $P(s_{qi}) = 1$.
- For a misclassification, $P(s_{qi}) = \frac{1}{100 conf(s_{qi})}$

3.3.4 Hidden Markov Models Based Technique (HMM)

Techniques that apply HMMs for modeling sequences, transform an input sequence from the symbol space to the hidden state space. The key assumption for the HMM based anomaly detection technique [8] is that the normal sequences can be effectively represented in the hidden state space, while anomalous sequences cannot be.

The training phase involves learning an HMM with σ hidden states, from the normal sequences in $\mathbf{S}$ using the *Baum Welch* algorithm.In the testing phase, the optimal hidden state sequence for the given input test sequence S_q is determined, using the *Viterbi* algorithm.For every pair of consecutive states, $\langle s^H_{qi}, s^H qi+1\rangle$, in the optimal hidden state sequence, the state transition matrix provides a likelihood score for transitioning from s^H_{qi} to s^H_{qi+1}. Thus a likelihood score vector of length $|S_q|-1$ is obtained.

The number of hidden states σ is a critical parameter for HMM. We experimented with values ranging from 2 to $|\Sigma|$. Our experiments reveal that the performance of HMM does not vary significantly for different values of σ. Here the results are presented for $\sigma = 4$.

3.4 Combining Scores

The window based and Markovian techniques discussed above generate a likelihood score vector for a test sequence, S_q. A combination function is then applied to obtain a single anomaly score $A(S_q)$. $A(S_q)$ can be computed in multiple ways, such as average score [15], minimum score, maximum score, average log score [24], using a threshold [19, 8]. We experimented with various combination functions for different techniques, and found that the *average log score* function has the best performance across all data sets. Hence, results are reported for the *average log score* function. If likelihood score for any window or symbol is 0, we replace it with 10^{-6} since $\log 0$ is undefined. Results with other combination techniques are available in our technical report [2].

4 Data Sets Used

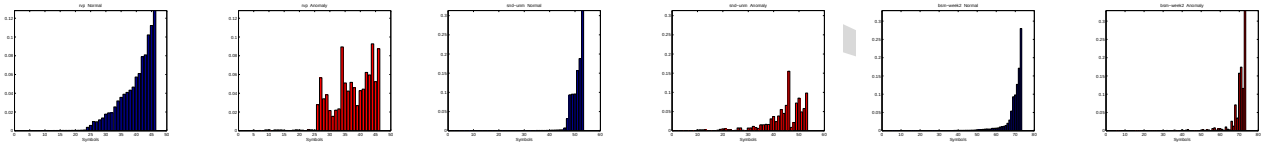
In this section we describe various public as well as the artificially generated data sets that we used to evaluate the different anomaly detection techniques. To highlight the strengths and weaknesses of different techniques, we also generated artificial data sets using HMMs. For every data set, we first constructed a set of normal sequences, and a set of anomalous sequences. A sample of the normal sequences was used as training data for different techniques. A disjoint sample of normal sequences and a sample of anomalous sequences were added together to form the test data. The relative proportion of normal and anomalous sequences in the test data determined the "difficulty level" for that data set. We experimented with different ratios such as 1:1, 10:1 and 20:1 of normal and anomalous sequences and encountered similar trends. In this paper we report results when normal and anomalous sequences were in 20:1 ratio in test data.

Results on data sets with other ratios are consistent in relative terms, although most techniques perform much better for the simplest data set that uses a ratio 1:1. In reality, the ratio of normal to anomalous can be even larger than 20:1. But we were unable to try such skewed distributions due to limited number of normal samples available in some of the data sets.

Source	Data Set	$\lvert\Sigma\rvert$	$\hat{l}$	$\lvert\mathbf{S^N}\rvert$	$\lvert\mathbf{S^A}\rvert$	$\lvert\mathbf{S}\rvert$	$\lvert\mathbf{S^T}\rvert$
PFAM	HCV	44	87	2423	50	1423	1050
	NAD	42	160	2685	50	1685	1050
	TET	42	52	1952	50	952	1050
	RUB	42	182	1059	50	559	525
	RVP	46	95	1935	50	935	1050
UNM	snd-cert	56	803	1811	172	811	1050
	snd-unm	53	839	2030	130	1030	1050
DARPA	bsmweek1	67	149	1000	800	10	210
	bsmweek2	73	141	2000	1000	113	1050
	bsmweek3	78	143	2000	1000	67	1050

Table 3: Public data sets used for experimental evaluation. $\hat{l}$ – Average Length of Sequences, $\mathbf{S^N}$ – Normal Data, $\mathbf{S^A}$ – Anomaly Data, $\mathbf{S}$ – Training Data, $\mathbf{S^T}$ – Test Data.

Table 3 summarizes the various statistics of the data sets used in our experiments. All data sets are available from our web site (Link not provided to maintain double blind review status). The distribution of the symbols for normal and anomalous sequences is illustrated in Figures 1(a),1(b) (RVP), 1(c),1(d) (snd-unm), and 1(e),1(f), (bsm-week2). The distribution of symbols in snd-unm data is different for



(a) RVP Normal (b) RVP Anomalous (c) snd-cert Normal (d) snd-cert Anomalous (e) bsm-w2 Normal (f) bsm-w2 Anomalous

Figure 1: Distribution of Symbols in Training Data Sets of Different Types.

normal and anomaly data, while the difference is not significant in RVP and bsm-week2 data. We will explain how the normal and anomalous sequences were obtained for each type of data set in the next subsections.

4.1 Protein Data Sets

The first set of public data sets were obtained from PFAM database (Release 17.0) [3] containing sequences belonging to 7868 protein families. Sequences belonging to one family are structurally different from sequences belonging to another family. We choose five families, viz., HCV, NAD, TET, RVP, RUB. For each family we construct a normal data set by choosing a sample from the set of sequences belonging to that family. We then sample 50 sequences from other four families to construct an anomaly data set. Similar data was used by [24] to evaluate the PST technique. The difference was that the authors constructed a test data for each pair of protein families such that samples from one family were used as normal and samples from the other were used as test. The PST results on PFAM data sets reported in this paper appear to be worse than those reported in [24].

4.2 Intrusion Detection Data Sets

The second set of public data sets were collected from two repositories of benchmark data generated for evaluation of intrusion detection algorithms. One repository was generated at University of New Mexico[2]. The normal sequences consisted of sequence of system calls generated in an operating system during the normal operation of a computer program, such as sendmail, ftp, lpr etc. The anomalous sequences consisted of sequence of system calls generated when the program is run in an abnormal mode, corresponding to the operation of a hacked computer. We report results on two data sets, viz, *snd-unm* and *snd-cert*.

The other intrusion detection data repository was the *Basic Security Module* (BSM) audit data, collected from a victim Solaris machine, in the DARPA Lincoln Labs 1998 network simulation data sets [18]. The repository contains labeled training and testing DARPA data for multiple weeks collected on a single machine. For each week we constructed the normal data set using the sequences labeled as normal from all days of the week. The anomaly data set was constructed in a similar fashion. The data is similar to the system call data described above with similar (though larger) alphabet.

The protein data sets and intrusion detection data sets are quite distinct in terms of the nature of anomalies. The anomalous sequences in a protein data set belong to a different family than the normal sequences, and hence can be thought of as being generated by a very different generative mechanism. This is also supported by the difference in the distributions of symbols for normal and anomalous sequences for RVP data as shown in Figures 1(a) and 1(b). The anomalous sequences in the intrusion detection data sets correspond to scenario when the normal operation of a system is disrupted for a short span. Thus the anomalous sequences are expected to appear like normal sequences for most of the span of the sequence, but deviate in very few locations of the sequence. Figures 1(e) and 1(f) shows how the distributions of symbols for normal and anomalous sequences in bsm-week2 data set, are almost identical. One would expect the UNM data sets (snd-unm and snd-cert) to have similar pattern as for the DARPA data. But as shown in Figures 1(c) and 1(d), the distributions are more similar to the protein data sets.

4.3 Artificial Data Sets

As mentioned in the previous section, the public data sets reveal two types of anomalous sequences, one which are arguably generated from a different generative mechanism than the normal sequences, and the other which result from a normal sequence deviating for a short span from its expected normal behavior. Our hypothesis is that different techniques might be suited to detect anomalies of one type or another, or both. To confirm our hypothesis we generate artificial sequences from an HMM based data generator. This data generator allows us to generate normal and anomalous sequences with desired characteristics.

We used a generic HMM, as shown in Figure 2 to model normal as well as anomalous data. The HMM shown in Figure 2 has two sets of states, $\{S_1, S_2, \ldots S_6\}$ and $\{S_7, S_8, \ldots S_{12}\}$.

Within each set, the transitions corresponding to the solid arrows shown in Figure 2 were assigned a transition probability of $(1-5\beta)$, while other transitions were assigned

[2]http://www.cs.unm.edu/∼immsec/systemcalls.htm

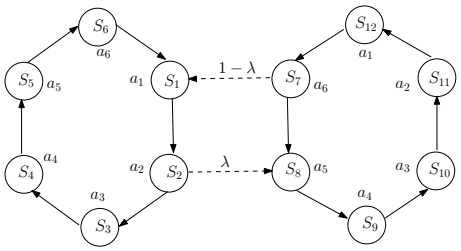


Figure 2: HMM used to generate artificial data.

transition probability β. No transition is possible between states belonging to different sets. The only exception are S_2S_8 for which the transition probability is λ, and S_7S_1 for which the transition probability is $1 - \lambda$. The transition probabilities S_2S_3 and S_7S_8 are adjusted accordingly so that the sum of transition probabilities for each state is 1.

The observation alphabet is of size 6. Each state emits one alphabet with a high probability $(1 - 5\alpha)$, and all other alphabets with a low probability (α). Figure 2 depicts the most likely alphabet for each state.

The initial probability vector π of the HMM is constructed such that either $\pi_1 = \pi_2 = \ldots = \pi_6 = 1$ and $\pi_7 = \pi_8 = \ldots = \pi_{12} = 0$; or vice-versa.

Normal sequences are generated by setting λ to a low value and π to be such that the first 6 states have initial probability set to $\frac{1}{6}$ and rest 0. If $\lambda = \beta = \alpha = 0$, the normal sequences will consist of the subsequence $a_1a_2a_3a_4a_5a_6$ getting repeated multiple times. By increasing λ or β or α, anomalies can be induced in the normal sequences.

This generic HMM can be tuned to generate two type of anomalous sequences. For the first type of anomalous sequences, λ is set to a high value and π to be such that the last 6 states have initial probability set to $\frac{1}{6}$ and rest 0. The resulting HMM is directly opposite to the HMM constructed for generating normal sequences. Hence the anomalous sequences generated by this HMM are completely different from the normal sequences.

To generate second type of anomalous sequences, the HMM used to generate the normal sequence is used, with the only difference that λ is increased to a higher value than 0. Thus the anomalous sequences generated by this HMM will be similar to the normal sequences except that there will be short spans when the symbols are generated by the second set of states.

By varying λ, β, and α, we generated several evaluation data sets (with two different types of anomalous sequences). We will present the results of our experiments on these artificial data sets in Section 7.

5 Evaluation Methodology

The techniques investigated in this paper assign an anomaly score to each test sequence $S_q \in \mathbf{S^T}$. To compare the performance of different techniques we adopt the following evaluation strategy:

1. Rank the test sequences in decreasing order based on the anomaly scores.

2. Count the number of true anomalies in the top p portion of the sorted test sequences, where $p = \delta q$, $0 \leq \delta \leq 1$, and q is the number of true anomalies in $\mathbf{S^T}$. Let there be t true anomalous sequences in top p ranked sequences.

3. Accuracy of the technique = $\frac{t}{q} = \frac{t}{\delta p}$.

We experimented with different values of δ and reported consistent findings. We present results for $\delta = 1.0$ in this paper.

Though computational complexity is an important metric when evaluating anomaly detection techniques in real application domains, we do not present a detailed comparison of computational complexity of different techniques due to space limitations. Briefly, since kernel based techniques involve computation of pairwise similarity, their computational complexity can be very high. The computation of similarity measure itself is a complex operation which can become a computational bottleneck for such techniques. Learning the model in HMM as well as RIPPER technique is expensive, though not as much as the computation of pairwise similarity for kernel based techniques. The PST technique is the most economical in terms of computational cost due to the pruning involved. t-STIDE, FSA, and FSA-z are relatively more expensive than PST though not very significantly. More detailed comparisons are provided in our technical report [2].

6 Nature of Normal and Anomalous Sequences

To understand the performance of different anomaly detection techniques on a given test data set, we first need to understand what differentiates normal and anomalous sequences in the test data set.

One distinction between normal and anomalous sequences is that normal test sequences are expected to be more similar (using a certain similarity measure) to training sequences, than anomalous test sequences. If the difference in similarity is not large, this characteristic will not be able to accurately distinguish between normal and anomalous sequences.

Another characteristic of a test sequence is the relative frequency of short patterns (subsequences) in the test sequence with respect to the training sequences. Let us classify a short pattern occurring in a test sequence as *seen* if it occurs in training sequences and *unseen* if it does not occur in training sequences. The *seen* patterns can be further

		Kernel			Markovian					
		cls	knn	tstd	fsa	fsaz	pst	rip	hmm	**Avg**
PFAM	hcv	0.54	0.88	0.90	0.88	0.92	0.74	0.52	0.10	0.69
	nad	0.46	0.64	0.74	0.66	0.72	0.10	0.20	0.06	0.45
	tet	0.84	0.86	0.50	0.48	0.50	0.66	0.36	0.20	0.55
	rvp	0.86	0.90	0.90	0.90	0.90	0.50	0.66	0.10	0.72
	rub	0.76	0.72	0.88	0.80	0.88	0.28	0.72	0.00	0.63
UNM	sndu	0.76	0.84	0.58	0.82	0.80	0.28	0.72	0.00	0.60
	sndc	0.94	0.94	0.64	0.88	0.88	0.10	0.70	0.00	0.64
DRPA	bw1	0.20	0.20	0.20	0.40	0.50	0.00	0.20	0.00	0.21
	bw2	0.36	0.52	0.36	0.52	0.56	0.10	0.18	0.02	0.33
	bw3	0.52	0.48	0.60	0.64	0.66	0.34	0.50	0.20	0.49
Avg		0.62	0.70	0.63	0.70	0.73	0.31	0.48	0.07	

Table 4: Results for public data sets.

classified as *seen-frequent*, if they occur frequently in the training sequences, and *seen-rare*, if they occur rarely in the training sequences. A given test sequence can contain all three type of patterns, in varying proportions. The performance of a window based or a Markovian technique will depend on following factors:

1. What is the proportion of *seen-frequent*, *seen-rare*, and *unseen* patterns for normal sequences, and for anomalous sequences?

2. What is the relative score assigned by the technique to the three type of patterns?

We will refer back to this characteristic when analyzing the performance of different techniques in Section 7.4.

7 Experimental Results

The experiments were conducted on a variety of data sets discussed in Section 4. The various parameter settings associated with each technique were explored. The results presented here are for the parameter setting which gave best results across all data sets, for each technique. The parameter settings for the reported results are : CLUSTER ($c = 32$), kNN ($k = 4$), FSA,FSA-z ($n = 5, l = 1$), tSTIDE ($k = 6$), PST ($L = 6, P_{min} = 0.01$), RIPPER ($k = 6$). For public data sets, HMM was run with $\sigma = 4$, while for artificial data sets, HMM was run with $\sigma = 12$. For window based and Markovian techniques, the techniques were evaluated using different combination methods discussed in Section 3.4.The results reported here are for the *average log score* combination function.

7.1 Results on Public Data Sets

Table 4 summarizes the results on 10 public data sets. Overall one can observe that the performance of techniques in general is better for PFAM data sets and on UNM data sets, while the DARPA data sets are more challenging. Though the UNM and DARPA data sets are both intrusion detection data sets, and hence are expected to be similar in nature, the results in Table 4 show that the performance on UNM data sets is similar to PFAM data sets. A reason for this could be that the nature of anomalies in UNM data sets are more similar to the anomalies in PFAM data sets. The similarity in the distribution of symbols for normal and anomalous sequences for PFAM and UNM data sets (See Figure 1), supports this hypothesis. CLUSTER and kNN show good performance for PFAM and UNM data sets but perform poorly on DARPA data sets. FSA and FSA-z show consistently good performance for all public data sets. t-STIDE performs well for PFAM data sets but its performance degrades for both UNM and DARPA data sets. While PST performs average to poor for all data sets, RIPPER performs well for UNM data sets.

7.2 Results on Artificial Data Sets

Table 5 summarizes the results on 6 artificial data sets. The normal sequences in data set $d1$ were generated with $\lambda = 0.01, \beta = 0.01, \alpha = 0.01$. The anomalous sequences were generated using the first setting as discussed in Section 4.3, such that the sequences were primarily generated from the second set of states. For data sets $d2$–$d6$, the HMM used to generate normal sequences was tuned with $\beta = 0.01, \alpha = 0.01$. The value of λ was increased from 0.002 to 0.01 in increments of 0.002. Thus normal sequences in $d2$ contain least number of anomalous patterns while those in $d6$ contain the largest number of anomalous patterns. The anomalous sequences were generated using the second setting in which λ is set to 0.1. One can observe in Table 5 that the data sets become progressively challenging from $d2$ to $d6$, as the performance of most of the techniques deteriorates as λ increases. The anomalous sequences are very different than normal for $d1$, since they are generated by a completely opposite generative mechanism, and hence all techniques are able to detect exactly all anomalous sequences.

From Table 5, we observe that PST is the most stable technique across the artificial data sets, while the deterioration is most pronounced for FSA and FSA-z. Both kNN and CLUSTER also get negatively impacted as the λ increases but the trend is gradual than for FSA-z.

	Kernel			Markovian					
	cls	knn	tstd	fsa	fsaz	pst	rip	hmm	**Avg**
d1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
d2	0.80	0.88	0.82	0.88	0.92	0.84	0.78	0.50	0.80
d3	0.74	0.76	0.64	0.50	0.60	0.82	0.64	0.34	0.63
d4	0.74	0.76	0.64	0.52	0.52	0.76	0.66	0.42	0.63
d5	0.58	0.60	0.48	0.24	0.32	0.68	0.52	0.16	0.45
d6	0.64	0.68	0.50	0.28	0.38	0.68	0.44	0.66	0.53
Avg	0.75	0.78	0.68	0.57	0.62	0.80	0.67	0.51	

Table 5: Results for artificial data sets.

7.3 Results on Altered RVP Data Set

Third set of experiments was conducted on the RVP data set from PFAM repository. A test data set was constructed

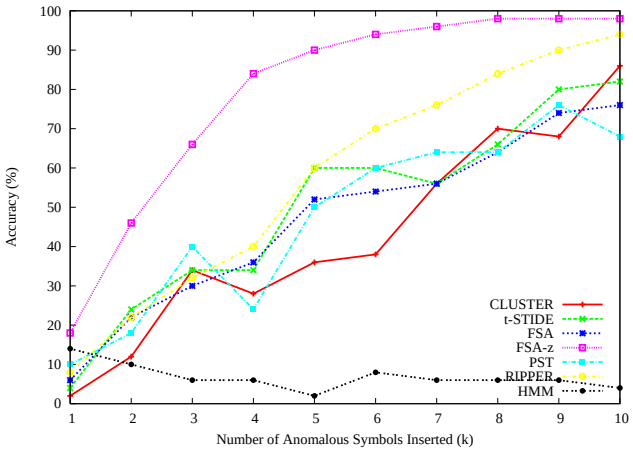


Figure 3: Results for altered RVP data sets

by sampling 800 most normal sequences not present in training data. Anomalies were injected in 50 of the test sequences by randomly replacing k symbols in each sequence with the least frequent symbol in the data set. The objective of this experiment was to construct a data set in which the anomalous sequences are minor deviations from normal sequences, as observed in real settings such as intrusion detection. We tested data sets with different values of k using CLUSTER, t-STIDE, FSA, FSA-z, PST, and RIPPER. Figure 3 shows the performance of the different techniques for different values of k from 1 to 10. We observe that FSA-z performs remarkably well for these values of k. CLUSTER, t-STIDE, FSA, PST, and RIPPER exhibit moderate performance, though for values of k closer to 10, RIPPER performs better than the other 4 techniques. For $k > 10$, all techniques show better than 90% accuracy because the anomalous sequences become very distinct from the normal sequences, and hence all techniques perform comparably well. Note that the average length of sequences for RVP data set is close to 90.

7.4 Observations for Individual Techniques

kNN The kNN technique performs better than CLUSTER for most of the data sets. This is expected, since CLUSTER is optimized for clustering and not for anomaly detection. The kNN based technique is not very sensitive to the parameter k for lower values of k, $1 \leq k \leq 4$, but the performance deteriorates for k larger than 4.

CLUSTER Performance of CLUSTER technique depends on the similarity measure. The similarity measure should be such that it assigns higher similarity between a pair of normal sequences, and lower similarity between a normal and anomalous sequence. If the anomalous sequence is very different from the normal sequence, $nLCS$ will assign a low similarity to that pair. But if the anomalous sequence is different from the normal sequence for a short span only, the $nLCS$ measure does not effectively capture this difference. In Table 4, CLUSTER performs well for protein data sets, as well as on UNM data sets, since the anomalous sequences are very different from the normal sequences. But the performance becomes worse for the DARPA data sets, such as bw1, where the anomalous sequences are minor deviations of the normal sequences. Results on artificial data sets (Table 5 and Figure 3) also indicate similar results.

FSA and FSA-z The original FSA technique does not assign any score when a unreachable state is encountered in the test sequence, while our FSA-z technique assigns a score of 0 in such cases. This generally improves the performance of FSA-z. The reason for this improvement is that often, mostly in anomalous sequences, there exist *unseen* patterns (Refer to our discussion in Section 6). While FSA ignores such patterns, FSA-z assigns a score of 0. Such patterns can provide useful information in differentiating between normal and anomalous sequences, and hence FSA-z performs better than FSA. In fact, FSA-z performs well if the anomalous test sequences contain relatively higher number of *unseen* patterns when compared to anomalous normal sequences. This behavior is evident in Figure 3, where insertion of even a few unseen symbols in the anomalous sequences results in multiple *unseen* patterns, and are easily detected by FSA-z, while FSA does not perform as well.

A drawback of FSA and FSA-z techniques is that they tend to assign high likelihood scores to *seen-rare* patterns. For example, let us assume that the subsequence AAAAAB occurs just once in training data, $\mathbf{S}$. A 5+1 FSA will learn this pattern and assign a probability of 1 if symbol B follows the subsequence AAAAA in a test sequence. Thus if the anomalous test sequences are such that they contain many *seen-rare* patterns, such anomalous sequences will be assigned a low anomaly score. The performance of FSA and FSA-z on artificial data set in Table 5 illustrates this point. For data sets $d2$–$d6$, the only difference between normal and anomalous sequences is that anomalous sequences contain higher number of *seen-rare* patterns when compared to normal sequences. But since FSA and FSA-z assign a high likelihood scores to such patterns, they fail to detect the anomalous sequences. This is the reason why performance of FSA as well as FSA-z deteriorates sharply from $d2$ to $d6$.

t-STIDE The t-STIDE technique has the best performance for most of the PFAM data sets but is relatively worse for the intrusion detection data sets. A strength of t-STIDE is that unlike FSA and FSA-z, t-STIDE does not get affected by the presence of *seen-rare* windows in anomalous test sequences. This can be observed in the artificial data sets, where the performance of t-STIDE does not dete-

riorate as sharply as for FSA-z as λ is increased (from $d2$ – $d6$).

The above mentioned strength of t-STIDE can also have a negative impact on its performance. t-STIDE learns only frequently occurring patterns in the training data, and ignores rarely occurring ones. Thus if a normal sequence is tested against t-STIDE, and it contains a *seen-rare* pattern, it will be assigned a low likelihood score by t-STIDE, while FSA and FSA-z will assign a higher score in such a scenario.

In the previous evaluation of t-STIDE [8] with other techniques, t-STIDE was shown to be comparable with RIPPER on UNM data sets. Our results are consistent with their evaluation. But on PFAM data sets we observe that t-STIDE performs better than RIPPER.

In the original t-STIDE technique [8] the authors use a threshold based combination function, in which the number of windows in the test sequence whose scores are equal to or below a threshold, are counted. Our experiments show that using the average log values of the scores performs equally well without relying on the choice of a threshold. The STIDE technique [12] is a simpler variant of t-STIDE in which the threshold is set to 0. In [8] the authors have shown that t-STIDE is better than STIDE, hence we evaluate the technique used in t-STIDE.

PST We observe that PST performs very poorly on most of the public data sets. It should be noted that in the paper that used PST for anomaly detection, the evaluation done on protein data sets was different than our evaluation. We provide a more unbiased evaluation which reveals that PST does not perform that well on similar protein data sets.

PST assigns moderately high likelihood scores to *seen-rare* patterns observed in a test sequence, since it computes the probability of the last event in the pattern conditioned on a suffix of the preceding symbols in the pattern. Thus the score assigned to *seen-rare* patterns by PST are lower than the score assigned by FSA-z, but higher than the score assigned by t-STIDE. Similarly, PST assigns a moderately high score for *unseen* patterns occurring in test sequences. The latter characteristic of PST can become the weakness of PST in a way that the anomaly signal due to the *unseen* as well as *seen-rare* patterns is *smoothed* by the PST. If in a data set the normal sequences contain a significant number of *seen-rare* patterns, and the anomalous sequences contain *unseen* patterns, they might still not be distinguishable. This behavior is observed for many public data sets in Table 4, as well as on the modified RVP data set in Figure 3.

The fact that the scores assigned to *seen-rare* patterns by PST are lower than the score assigned by FSA-z, can also become strength of PST because it does not get affected by the presence of *seen-rare* patterns in normal test sequences, unlike FSA-z. This characteristic is observed with the artificial data sets in Table 5. As λ increases, the performance of PST remains more stable than any other technique.

RIPPER Both RIPPER and PST techniques are more flexible than FSA-z in conditioning the probability of an event based on its preceding symbols. This suggests that RIPPER technique also *smoothens* the likelihood scores of *unseen* as well as *seen-rare* patterns in test sequences. Therefore, we observe that for public data sets, RIPPER exhibits relatively poor performance, in comparison to FSA and FSA-z. RIPPER performs better than PST on 8 out of 10 data sets. Thus choosing a sparse history is more effective than choosing a variable length suffix. The reason for this is that the smoothing done by RIPPER is less than smoothing done by PST (since the RIPPER classifier applies more specific and longer rules first, and is hence biased towards using more symbols of the history), and hence RIPPER is more similar to FSA and FSA-z than PST. The results on artificial data sets in Table 5 also show that the deterioration of RIPPER's performance from $d1$ – $d6$ is less pronounced than for FSA-z while more pronounced than PST.

HMM The HMM technique performs very poorly on all public data sets. The reasons for the poor performance of HMM are twofold. The first reason is that HMM technique makes an assumption that the normal sequences can be represented with σ hidden states. Often, this assumption does not hold true, and hence the HMM model learnt from the training sequences cannot emit the normal sequences with high confidence. Thus all test sequences (normal and anomalous) are assigned a low probability score. The second reason for the poor performance is the manner in which a score is assigned to a test sequence. The test sequence is first converted to a hidden state sequence, and then a $1 + 1$ FSA is applied to the transformed sequence. We have observed from our experiment using FSA that a $1 + 1$ FSA does not perform well for anomaly detection. The performance of HMM on artificial data sets (Table 5 illustrates this argument. Since the training data was actually generated by a 12 state HMM and the HMM technique was trained with $\sigma = 12$; thus the HMM model effectively captures the normal sequences. The results of HMM for artificial data sets are therefore better than for public data sets, but still slightly worse than other techniques because of the poor performance of the $1 + 1$ FSA.

8 Conclusions and Future Work

Our experimental evaluation provided us with valuable insights into strengths and weaknesses of different anomaly detection techniques. None of the techniques was found to be consistently superior to all other techniques, indicating that the performance of a technique depends on the nature of the sequence data set. The use of artificial data generator allowed us to arrive at conclusions that were not evident

from the results on public data sets.

A significant result of this study is that several techniques have been shown to be quite effective in application domains for which they were not originally intended for. Techniques such as t-STIDE and FSA ,which were originally evaluated on system call intrusion detection data, show promising results on protein data sets. Interestingly, t-STIDE performs relatively poorly on system call intrusion detection data sets.

Results on the public data sets (Table 4) reveal that FSA-z and FSA, are the most consistent techniques while PST and RIPPER generally perform poorly. But the results on artificial data sets (Table 5) identify scenarios where the latter two techniques might be better suited than the former two.

Kernel based techniques are found to perform well for data sets in which the anomalous sequences are relatively different from the normal sequences; but perform poorly when the different between the two is small. This is due to the nature of the normalized LCS similarity measure used in the kernel based techniques. Future work should investigate other similarity measures that are able to capture the difference between sequences that are minor deviations of each other. Our experiments show that kNN technique is somewhat better suited than CLUSTER for anomaly detection.

Consistent with the observations of other researchers [8], we found the HMM technique to perform poorly. When the normal sequences were generated using an HMM, the performance improves significantly. The hidden state sequences, obtained as a intermediate transformation of data, can actually be used as input data to any other technique discussed here. The performance of such an approach will be investigated as a future direction of research.

9 Acknowledgements

This work was supported by NASA under award NNX08AC36A. Access to computing facilities was provided by the University of Minnesota Digital Technology Center and Supercomputing Institute.

References

[1]

[2] Comparing anomaly detection techniques for sequence data. Details suppressed to maintain blind review status.

[3] A. Bateman, E. Birney, R. Durbin, S. R. Eddy, K. L. Howe, and E. L. Sonnhammer. The pfam protein families database. *Nucleic Acids Res.*, 28:263–266, 2000.

[4] S. Budalakoti, A. Srivastava, R. Akella, and E. Turkov. Anomaly detection in large sets of high-dimensional symbol sequences. Technical Report NASA TM-2006-214553, NASA Ames Research Center, 2006.

[5] W. W. Cohen. Fast effective rule induction. In A. Prieditis and S. Russell, editors, *Proceedings of the 12th International Conference on Machine Learning*, pages 115–123, Tahoe City, CA, jul 1995. Morgan Kaufmann.

[6] W. L. E. Eskin and S. Stolfo. Modeling system call for intrusion detection using dynamic window sizes. In *Proceedings of DISCEX*, 2001.

[7] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff. A sense of self for unix processes. In *Proceedinges of the ISRSP96*, pages 120–128, 1996.

[8] S. Forrest, C. Warrender, and B. Pearlmutter. Detecting intrusions using system calls: Alternate data models. In *Proceedings of the 1999 IEEE ISRSP*, pages 133–145, Washington, DC, USA, 1999.

[9] B. Gao, H.-Y. Ma, and Y.-H. Yang. Hmms (hidden markov models) based on anomaly intrusion detection method. In *Proceedings of International Conference on Machine Learning and Cybernetics*, pages 381–385, 2002.

[10] F. A. Gonzalez and D. Dasgupta. Anomaly detection using real-valued negative selection. *Genetic Programming and Evolvable Machines*, 4(4):383–403, 2003.

[11] V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

[12] S. A. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. *Journal of Computer Security*, 6(3):151–180, 1998.

[13] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., 1988.

[14] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of SIAM International Conference on Data Mining*. SIAM, May 2003.

[15] W. Lee and S. Stolfo. Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, 1998.

[16] W. Lee, S. Stolfo, and P. Chan. Learning patterns from unix process execution traces for intrusion detection. In *Proceedings of the AAAI 97 workshop on AI methods in Fraud and risk management*, 1997.

[17] C. S. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for svm protein classification. In *Pacific Symposium on Biocomputing*, pages 566–575, 2002.

[18] R. P. Lippmann and et al. Evaluating intrusion detection systems - the 1998 darpa off-line intrusion detection evaluation. In *DARPA Information Survivability Conference and Exposition (DISCEX) 2000*, volume 2, pages 12–26, 2000.

[19] C. C. Michael and A. Ghosh. Two state-based approaches to program-based anomaly detection. In *Proceedings of the 16th Annual Computer Security Applications Conference*, page 21. IEEE Computer Society, 2000.

[20] Y. Qiao, X. W. Xin, Y. Bin, and S. Ge. Anomaly intrusion detection method based on hmm. *Electronics Letters*, 38(13):663–664, 2002.

[21] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 427–438. ACM Press, 2000.

[22] D. Ron, Y. Singer, and N. Tishby. The power of amnesia: learning probabilistic automata with variable memory length. *Machine Learning*, 25(2-3):117–149, 1996.

[23] A. N. Srivastava. Discovering system health anomalies using data mining techniques. In *Proceedings of 2005 Joint Army Navy NASA Airforce Conference on Propulsion*, 2005.

[24] P. Sun, S. Chawla, and B. Arunasalam. Mining for outliers in sequential databases. In *In SIAM International Conference on Data Mining*, 2006.

[25] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.